6-2005

A Perception Based, Domain Specific Expert System for Question-Answering Support

Raheel Ahmad
Southern Illinois University Carbondale

Shahram Rahimi
Southern Illinois University Carbondale, rahimi@cs.siu.edu

Published in Ahmad, R., & Rahimi, S. (2005). A perception based, domain specific expert system for question-answering support. Annual Meeting of the North American Fuzzy Information Processing Society, 2005. NAFIPS 2005, 454-459. doi: 10.1109/NAFIPS.2005.1548578

A Perception Based, Domain Specific Expert System for Question-Answering Support

Raheel Ahmad, Shahram Rahimi
Department of Computer Science
Southern Illinois University
Carbondale, Illinois 62901
[rahmad, rahimi]@cs.siu.edu

***Abstract*: The ability to search has become an integral part of our interaction with technology. However, the current search technologies mostly use a keyword based searching mechanism, which does not have any deductive abilities. The recent introduction of the idea of Computing with Words, which can give deduction abilities to existing technologies, provides a new base for developing frameworks. This paper proposes implementation of a domain specific Fuzzy Expert System based on a question-answer system, which employs Computing with Words. In order to perform the translation of natural language sentences into a standard format, Probabilistic Context-Free Grammar is used.**

I. INTRODUCTION

Due to technological advances, users have access to a large amount of information. However, in order for a user to utilize this data effectively, it is often necessary to provide an assisted system for knowledge management; such as a search engine. A search engine has two services that it must provide: first, it must understand the user's needs according to a given query, and second, the search engine must perform actual searching within the particular knowledge domain. Therefore, one of the basic necessities for a search engine to be successful is the ability to *perceive* the user's needs. The development of a perceptive system must be centered around using natural language since this is the most effective means through which humans are able to express and understand thoughts and ideas. Developing such a perception-aware system has been particularly difficult, and it involves seemingly disparate fields such as computer science and linguistics. Within computer science itself, it requires the expertise of almost every field of research. Because of these difficulties, a perception-aware system is still not a reality.

The idea of using natural languages as the communication channel between humans and machines is not new. However, the use of natural language in computers has been in general superficial and has resulted in systems that only partially interact in natural language with a user, while the systems' inner workings lack any overall support for the intricacies of a natural language. This is not for lack of trying, though, as implementing an artificial system that supports natural language is a rather difficult process. However, the use of fuzzy logic to support natural language interaction between a user and a system seems promising. In order for systems to utilize the more intuitive fuzzy logic they would have to change the bivalent logic they currently use. This would of course require a complete overhaul of the way we develop such systems currently.

This paper gives a development plan for a domain-specific system that answers natural language queries, based on Fuzzy Expert Systems. The proposed system supports true question-answering between a human and a computer. The core of this system is based on Zadeh's introduction of the concept of Computing with Words (CwW) and Perceptions [1], which is founded on fuzzy set theory and fuzzy inference rules. The concept of performing computations directly on "perceptions", which are expressed using "words", is a marked shift from the current trend in computing. This paper is intended to be a step towards the advent of such technology which would hopefully lead to computing machines which are much easier to interact with.

One of the key ideas presented in this paper is the use of Probabilistic Context-Free Grammar (PCFG) for interfacing the fuzzy logic of CwW with natural language. This is one of the two areas that need to be dealt with when developing a framework for computing with words, the second area being finding rules for inferencing. PCFG provides a good quality solution for translating natural language perceptions into the standard formats that Zadeh has introduced for CwW. The other key idea in this paper is to use a fuzzy expert system to implement the storing of translated perceptions, and to perform the inferencing that is required for deduction when a query is presented to the system.

The remainder of this paper is organized in the following manner: section II provides the required background pertaining to the framework of the proposed expert system. This section includes an introduction to PCFG; its use in general and in relation to natural language, and an overview of its place in CwW in respect to the framework. Also, the details of CwW which are needed for this paper are discussed briefly. Section III presents the architecture of the proposed expert system, which includes a more detailed description of the methodology used in the system. Section IV mentions the future of this work, along with additional comments and conclusions.

0-7803-9187-X/05/$20.00 ©2005 IEEE.

II. BACKGROUND

A. Computing with Words - Main ideas

The idea of computing with words redefines how we interact with computers, or rather how computers understand us. In order to perform such computations we use the concept of *perceptions*. *Perceptions* in natural language consist of a fuzzy constraint on a variable. A perception's constraint and the constrained variable are then *explicated*, and the translated perception is produced in the form of a *protoform* (*proto*typical *form*). A protoform is a standard way of representing constraints in CwW. For example, a simple natural language statement (perception):

Traffic is heavy.

consists of a constraint *heavy* on the variable *Traffic*. This constraint is in the form represented by 'X is R', where the variable X is for the noun *Traffic*, and the constraint R is for the adjective *heavy*. The 'is' in the protoform is different from the 'is' in the English statement and is used to represent the constraint as a possibility distribution of the variable. [ZadehPaper2]. Several such protoforms are specified which represent different ways in which the constraint is applied.

The generalized constraint is represented as:

$$X \text{ isr } R$$

where expanding the 'r' in 'isr' gives various specific constraints such as the following:

X is R	(disjunctive)
X ise R	(equality)
X isp R	(probability)
X isu R	(usuality)
X isv R	(veristic)

One of the key problems when using CwW is translating perceptions given in natural language into a standard format that can be manipulated with computation. If a set of documents is available in natural language, it is necessary to translate these documents into a standard format before we can perform any deduction based on the implicit perceptions on these documents. *The* translating mechanism should recognize the syntax of the natural language propositions in order to translate them into a GCL format. We propose using Probabilistic Context-Free Grammar, introduced in the next section, to perform this translation.

B. Probabilistic Context Free Grammar

A context free grammar (CFG) describes a language by providing a set of production rules which govern how a non-terminal symbol of the language can be expanded into a set of terminal and non-terminal symbols. The CFG forms part of the *phase structure (PS) grammars* which were introduced by N. Chomsky [3] in 1957 when he applied Post production rules to natural languages. A production rule for a context-free grammar is of the form $S \rightarrow w$, where S is the language symbol and w represents a string of terminal symbols (words) or non-terminal symbols. The production rules are classified as either *terminal* or *structural* rules. Terminal rules have a word in the right hand side, which of course cannot be expanded, while structural rules have non-terminal rules on both sides of the rule.

Probabilistic CFG (PCFG) associates a probability with each production rule, with the sum of the probabilities for rules with the same left hand side being unity. The probability of a sentence parsed by a set of rules is the product of the probabilities of all the involved rules. This is illustrated in Fig. 1.

The sentence, "the p b n" has two different parsing possibilities with the same grammar. The computed probability is calculated for the two parse trees and the parse tree with more probability is considered the proper or preferred inference. This process is called syntactic disambiguation.

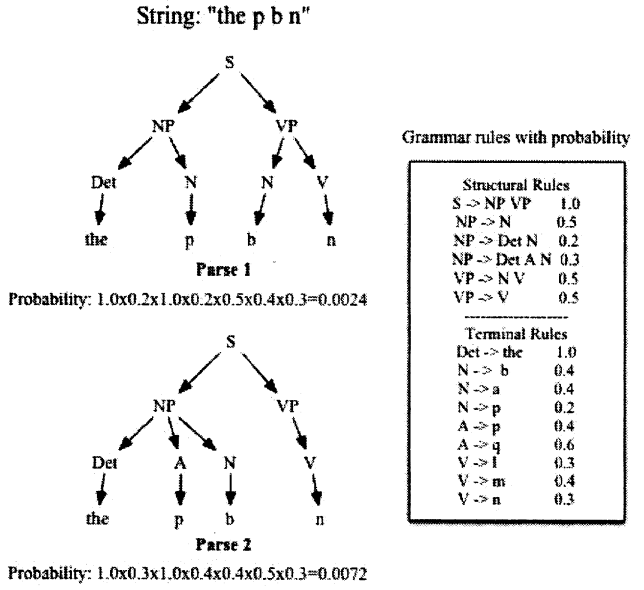

Fig. 1 Parsing the same sentence in two different ways

III. A FUZZY EXPERT SYSTEM FOR QUESTION-ANSWERING

This section describes the idea and architecture of an Expert System based Query Support mechanism, which can provide question-answering support for a particular domain. The system is conceptually divided into two modules - a pre-processing module, which builds the fuzzy expert system for the domain, and the actual query system, which allows a user to ask questions about the domain in natural language and provides answers using the expert system. This system is being implemented using Fuzzy Clips, which is a fuzzy logic extension to the popular Clips expert system [4].

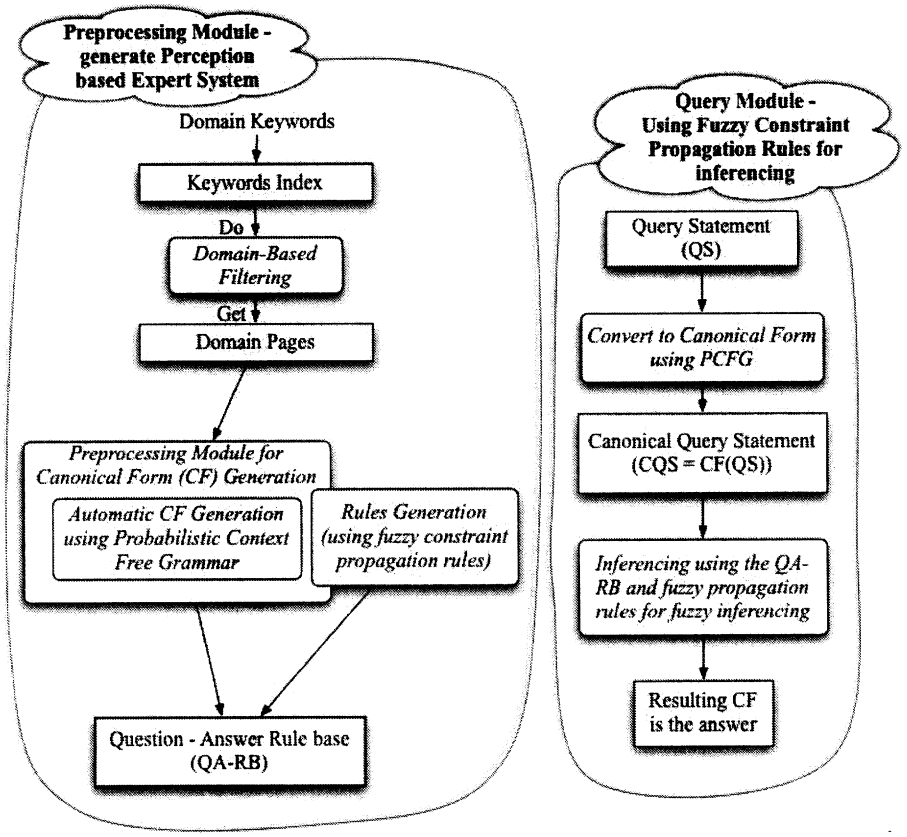

Fig. 2 Architecture of the Question-Answering system

The difference between a conventional search engine and a question-answer system must be made clear. A search engine has a limited functionality and usefulness. It traditionally finds certain keywords within a given query and searches for these keywords in its preprocessed database and provides the user with a list of documents. A question-answering (QA) system allows the user to query the system for knowledge using a restricted natural language syntax where the user expects an answer to the query rather than a list of documents that contain the query keywords. In the current proposal we have restricted the effort to a single bounded domain. Doing this is necessary as parts of the whole process rely heavily on automatic knowledge extraction and inference, and the implementation of such components in an unbounded domain would require much more effort.

A question-answer system offers several advantages over traditional keyword based searching. It is much more intuitive to the user, who simply formulates a question instead of making up a set of keywords. The QA system returns more relevant results, since it attempts to understand what the user wants; in other words, it understands human perception regarding the query, albeit in a limited manner. In the traditional search engines, upon the return of the query results, users still have to browse through the resulted documents and analyze them in order to come at a conclusive answer. A QA system does not require this extra step, since it has deductive capabilities which allow it to directly "answer" the user's query. Below we describe the details of the above components one by one.

A. Preprocessing Components

This component creates the *rule base* for the QA Expert System. This is an offline process, similar to the keyword indexing done in a conventional search engine.

The initial stage of the preprocessing requires filtering of the documents that are available in the domain. The search returns a set of documents that are relevant to the domain. This is accomplished by using a set of domain keywords, which is a collection of all the keywords representing the domain [10]. Traditionally this filtering can be performed by the following methods: a) using an expert's opinion in deciding the relevancy of a document to the domain, b) by calculating the occurrence of the domain keywords in every document [11] c) Using the Term Frequency Inverse Document Frequency (TFIDF) criterion [12] d) document classification using the Bayesian classifier [13].

After obtaining the documents related to the domain using the above filtering process, the natural language text needs to be translated into the standard canonical form for CwW to operate. This is done by the Canonical Form Generation module shown in Fig. 2. The PCFG system introduced in the previous section plays a major role in this module, which is discussed next.

B. Using PCFG for translating Natural Language to GCL

Performing language translation is an extremely difficult process and is almost impossible to do a perfect unsupervised natural language translation, either to a formal language or another natural language. There are several issues including syntax, semantics, morphology, and lexicon, which must be taken into consideration when considering a translation. For this purpose, PCFG provides an easy to use tool to describe the grammar of a language, which can be employed in doing the translation from natural language to GCL.

In order to perform the translation, we propose forming a set of production rules for each of the above recognized constraints. Therefore a set of production rules for each canonical constraint can be defined for a particular domain, according to the various forms that a constraint takes in natural language. This can be then seen as a mapping of perceptions from a natural language to a standard canonical form. Clearly, such a set of rules may differ for the same canonical form from one domain to another. Theoretically, such a set of rules could be defined for a particular language with no domain limitation.

For example, for the (most popular) constraint form - *X is R*, which represents a possibility distribution of the relation R, the following production rules may be used to define the PCFG:

Structural Rules:

LHS		***RHS***	***Probability***
S	->	NP conP DP	1.0
conP	->	is	0.7
conP	->	are	0.3
NP	->	NP Con NP	0.2
NP	->	Noun	0.5
NP	->	Det NP	0.3
DP	->	AP	0.5
DP	->	negP AP	0.2
DP	->	QP AP	0.1
DP	->	negP QP AP	0.2

Terminal Rules:

Noun	->	John	0.4
Noun	->	Dave	0.2
Noun	->	Mary	0.4
Con	->	and	0.6
Con	->	with	0.4
Det	->	a	0.3
Det	->	the	0.7
negP	->	not	0.6
negP	->	never	0.4
AP	->	hungry	0.2
AP	->	present	0.4
AP	->	late	0.4
QP	->	very	0.3
QP	->	almost	0.7
etc..			

The above structural and terminal rules combined give a grammar that can be used to recognize several natural language perceptions of the form *X is R*, for example:

Lisa is hungry.
John is never late.
John and Dave are almost late.

When examined by a human, the above sentences appear to be in the correct canonical form, while but this may not be immediately apparent to a computer. A *push-down automaton* can be used to recognize a context free language. As an *acceptor,* it can recognize whether a given sentence is generated by a given CFG and as a *generator* [5]. For a probabilistic CFG, parsing techniques have been proposed which provide the parse of a sentence with the highest parsing probability [7, 8].

Although there may be many PCFGs for a target domain, there exist only a single set of terminal rules for all of them. However, the structural rules for each PCFG would be different. The details of the preprocessing module are discussed below.

1) *Generation of Terminal Rules*: For a particular domain, we collect a set of all *domain keywords.* These keywords define information that is relevant to the domain and provide the terminal rules for the PCFGs. Although it is not trivial to automate terminal rules' generation, there are certain methods that may be adopted for this purpose [6]. Nonetheless, the best approach at this moment would be to have an expert specifying the terminal rules.

As mentioned earlier, it must be noted that the terminal rules remain the same for all the PCFGs that are written for the various canonical forms. The terminal rules are a part of the general domain knowledge and are shared by all the canonical representations in the grammar.

2) *Canonical Form discovery using PCFG*: This is the most important part of the preprocessing phase. In this module the sentences of the available documents are transformed into perceptions in canonical form. This can be done by matching each sentence to a grammar for a particular canonical form. There are various algorithms that are currently present to provide this matching, one of which is using unsupervised learning [8]. This method uses already available hand-parsed data for calculating parsing probabilities for individual rules. Such hand-parsed training data may not be very difficult to construct for a specific domain, and provides the most efficient and accurate parser.

A particular sentence can be generated in more than one way (more than one parse tree) by a particular PCFG. In this case, the sum of the probabilities of all such parse trees gives the probability of that PCFG having generated that sentence. Moreover, a sentence can be found to be generated by more than one grammar using different parse trees. The probabilities of all parse trees for a PCFG that generated the sentence are summed together to give the cumulative probability of the PCFG. The PCFG with the highest such probability is considered to be the canonical form that represents the implicit constraint of this natural language perception.

3) *Facts Generation*: In the previous step, the natural sentences are mapped to their related canonical form. In this step, canonical forms are stored as facts in the fact base of the

expert system. This is a fairly painless process since the canonical forms are as close in representation to a fact as required by the expert system. For example, the canonical form, *X is R*, can be represented by the deftemplate:

```
(deftemplate XisR "Form: John is not late"
    (slot Noun)
    (slot conP)
    (slot negP)
    (slot AP)
)
```

Suppose the sentence *"John is not angry"* is recognized correctly as a sentence generated by the above grammar and the proper constituents of the sentence are instantiated according to the deftemplate above. At this stage, the expert system does not have to realize the functional mapping of these constituents to the form *X is R*. This would be the job of the inference rules, which form the inference engine, to identify the components of the canonical form while performing the inferencing.

After generating the perceptions in canonical form by discovering the its matching PCFG, the respective deftemplate for that PCFG is instantiated for each such perception. The perceptions are the facts asserted into the fact base, which is the 'working memory' of the expert system.

4) *Rules Generation*: This step is the basis for the formation of the expert system. The rule base of the perception-based expert system is the main processing unit of the query system. These rules determine the 'answer' of the system given a certain query in the form of a question. Following the core ideas of computing with words, these rules include the major fuzzy propagation rules specified by Zadeh [2], which are comprised mainly of fuzzy inference rules. Some of the more important rules are listed in Table I.

These rules are currently being implemented in Fuzzy Clips. However, there are several issues that need to be addressed when implementing these propagation rules for perceptions in our expert system. These issues can be exemplified by considering the sentence *Most airplanes have big wings*. This sentence can be easily understood by a simple grammar to be of the form *X is R*. However, when we come across a question such as *"What is the size of an airplane's wing?"*, the following details are to be handled while implementing the rules:

- Define 'most' and 'big' in the form of fuzzy sets. For example, 'big' has to be defined for size(airplane(wing)), i.e., size of an airplane's wing, by using available data for the wing's size with corresponding membership values of the adjective 'big'. This assumption of having readily available data to derive such a definition may not always be true.

TABLE I
EXAMPLES OF FUZZY CONSTRAINT PROPAGATION RULES

Conjuction Rule *X* is *A* *X* is *B* *X* is $A \cap B$ *Disjunction Rule* *X* is *A* *X* is B *X* is $A * B$ *Compositional rule for probabilistic and possibilistic constraints* *X isp R* *(X; Y) is S* *Y isrs T; (4.31)*	*Compositional Rule* *X* is *A* *(X, Y)* is B *Y* is $A \circ B$ *Extension Principle* *X* is *A* *f(X)* is *f(A)* *Generalized Extension Principle* *f(X)* is *A* *q(X)* is $q(f^{-1}(A))$

- Relate the adjective 'big' to the noun 'size'. Such a relation should be defined for all possible combinations of nouns and adjectives, verbs and adverbs, etc.
- Calculation of 'size'. The noun 'size' has to be mathematically calculated from the available perceptions and there should be a definition available beforehand to perform such a calculation.

Zadeh offers a similar example when explaining the usage of the propagation rules [2]. Assumptions are made regarding the availability of an 'explanatory database' which can solve the problem a. However, problems b and c still remain and are left as a detail to be solved by a particular implementation.

There are several other issues need to be tackled, including morphology and syntactic disambiguation, before we can comprehensively implement the constraint propagation rules. Nonetheless, the expert system model is an ideal fit for stating these rules.

C. Query Component

This component performs the query processing and is responsible for answering a given question.

1) Converting query to a Canonical Form: This step is similar to step 2 of the preprocessing module. This involves changing the query string, which should be entered in a restricted natural language format, into a canonical form. It must be noted that we are not concerned with making the input query exactly like a natural language question. Therefore, the query does not necessarily have to be written for instance as *"What is the nearest town to Carbondale?"* for our system to be qualified a question-answering system. The main characteristic of a question –answering system is to *understand* the query and reply accordingly. Even so, the idea of using computing with words is to interface natural language with computing, so the eventual goal is to use a natural

language that is as general as possible. However, the current technology in natural language translation has not matured enough to allow such an interface.

After the translation to the canonical form, the query string is transformed to a fact through a process similar to that in step 3 of the preprocessing. This query fact is given to the expert system for the actual querying to take place.

2) Inferencing using the Rule Base: The rule base constructed by the preprocessing module is used in this step to provide the "answer" to the query in canonical from the last step. The question acts as a trigger for the expert system. The fuzzy constraint propagation rules, the expert system's rule base, process the request using its fact base (working memory) and provides a resulting fact as the answer to the query.

IV CONCLUSIONS AND FUTURE WORK

Fuzzy logic has proved to be exceptionally valuable and indispensable for various practical applications. However, its most important contribution to technology may come in the future, in the form of widespread use of the theory of Computing with Words. CwW has the potential to change the way we interact with computers and how computers perceive human needs. However, there is still a long way for this technology to mature and to be adopted by other technologies such as search engines.

In this paper, we have proposed an architecture for a search engine that exploits the unique ability of CwW to manipulate perceptions and to work with natural language. The use of Probabilistic Context-Free Grammar as an interface to CwW seems to be an appropriate fit as this interface provides an ability to analyze the structure of a sentence for generating its canonical form. The use of fuzzy expert system in this architecture provides a mechanism for implementation of the fuzzy constraint propagation rules for inferencing. The implementation of this expert system requires a deep understanding of linguistics. We are currently taking the preliminary steps for implementing the system by writing a domain-specific grammar, which should be able to handle most of the possible sentence structures as well as the required protoforms. As a future work, it would be interesting to study the use of the Generalized Fuzzy Context-Free Grammar [9] in place of PCFG to provide the interface to CwW.

REFERENCES

[1] L. A. Zadeh, "Fuzzy Logic = Computing with Words", *IEEE Trans. On Fuzzy Systems*, 1996, pp. 103-111.

[2] L. A. Zadeh, "From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Measurement of Perceptions", *IEEE Transactions on Circuits and Systems*, 1999, 45(1), pp. 105-119.

[3] Noam Chomsky, "Three models for the description of language", IRE Transactions on Information Theory, 2, 1956, pp.113-124.

[4] J. Hetherington, "Fuzzy Clips: A Rule Based Teaching Tool", *Processing of the 17th NACCQ*, 2004.

[5] H. Lewis, and C. Papadimitriou. *Elements of the Theory of Computation*, 1998, Upper Saddle River, NJ, Prentice Hall.

[6] D. V. Pynadath and M. P. Wellman, "Generalized Queries on Probabilistic Context-Free Grammars", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, 1998.

[7] Z. Chi, "Statistical Properties of Probabilistic Context-Free Grammar", in *Association for Computational Linguistics*, Vol. 25, No. 1, 1999.

[8] E. Charniak, "Statistical Parsing with Context-Free Grammar and Word Statistics", in *American Association for Artificial Intelligence*, 1997.

[9] P.R.J. Asveld, "Fuzzy context-free languages -- Part I: generalized fuzzy context-free grammars", *CTIT Tech. Rep. No. 00-03*, Enschede, the Netherlands, 2000.

[10] M. Chau and H. Chen, "Using Content-Based and Link-Based Analysis in Building Vertical Search Engines", *ICADL 2004*, Z. Chen et al. (Eds.). Springer-Verlag Berlin Heidelberg, LNCS 3334, pp. 515–518, 2004.

[11] J. Cho, H. Garcia-Molina, and L. Page, "Efficient Crawling through URL Ordering", in *Proceedings of the 7th International World Wide Web Conference*, Brisbane, 1998.

[12] O. Baujard, V. Baujard, S. Aurel, C. Boyer, and R. Appel, "Trends in Medical Information Retrieval on the Internet," *Computers in Biology and Medicine*, 28, 589–601, 1998.

[13] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced Hypertext Categorization Using Hyperlink", in *Proceedings of ACM SIGMOD International Conference on Management of Data*, Seattle, 1998.